We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

9

Fast Visual Tracking of Mobile Agents *

Andelko Katalenic[1], Ivica Draganjac[2], Alan Mutka[3] and Stjepan Bogdan[4]
[1]*Eindhoven University of Technology*
[2,3,4]*Faculty of Electrical Engineering and Computing University of Zagreb*
[1]*The Netherlands*
[2,3,4]*Croatia*

1. Introduction

Control and coordination of agents in multi-agent systems is demanding and complex task. Applications like search and rescue Jennings et al. (1997), mapping of unknown environments or just simple moving in formation demands gathering a lot of information from surrounding. A local sensor-based information is preferred in large formations due to observability constraints. On the other hand, local information processing requires sophisticated and expensive hardware that would manage to gather and process sensor information in real-time. Furthermore, refined sensors, such as a camera, entail time consuming algorithms. Generally, it is difficult to provide satisfactory results by implementation of such algorithms in real-time. For this reason immense research efforts have been put in development of fast and simple image analysis methods. This is specifically noticeable in the field of robot vision, a discipline that strives to solve problems such as robot localization and tracking, formation control, obstacle avoidance, grasping and so on. Visual tracking of robots in formation is usually based on visual markers. Algorithms for detection of predefined shapes or colors are simple enough to be executed even on low-cost embedded computers in real-time. In Chiem & Cervera (2004) pose estimation based on the tracking of color regions attached to the robot is presented. The position of the leader robot is estimated at video rate of 25 frames per second. The main disadvantage of the proposed method is marker position on the robot, i.e. a marker can be recognized only from particular angle. Authors in Cruz et al. (2007) accomplish robot identification and localization by using visual tags arranged on the back of each robot on a 3D-truncated octagonal-shaped structure. Each face of visual tag has a code that provides the vehicle's ID as well as the position of the face in the 3D-visual marker. This information allows a vision sensor to identify the vehicle and estimate its pose relative to the sensor coordinate system. A robot formation control strategy based on visual pose estimation is presented in Renaud et al. (2004). Robots visual perception is enhanced by the control of a motorized zoom, which gives the follower robot a large field of view and improves leader detection. A position-based visual servo control strategy for leader-follower formation control of unmanned ground vehicles is proposed in Dani et al. (2009). The relative pose and the relative velocity are obtained using a geometric pose estimation technique and a nonlinear velocity estimation strategy. The geometric pose estimation technique Gans (2008)

*This article is extended version of the paper originally presented at The 4th IEEE Conference on Industrial Electronics and Applications (ICIEA 2009)

uses Euclidean homography relationships and a single known geometric length on a single following object.

Visual detection of an object without usage of additional markers might be very difficult. In some cases, it is important to recognize and localize an object that has to be picked up or manipulated (stationary object), while in other cases, it is paramount to identify and pinpoint an object that needs to be followed (object in motion). Object detection algorithms are usually based on matching captured images with stored object models. Tomono Tomono (2006) proposed a high density indoor map-based visual navigation system on the basis of on-line recognition and shape reconstruction of 3D objects, using stored object models. However, the implemented Scale-invariant feature transform (SIFT) extraction is compute-intensive, and a real-time or even super-real-time processing capability is required to provide satisfactory results.

Currently, the fastest object detection is provided by cascades of classifiers based on Haar-like features Viola & Jones (2001). Apart from low execution times, the statistical nature of an this algorithm based on Haarlike features is robust to motion blur and changing lighting conditions because those features represent basic elements of the picture including edges, lines and dots. The cascades are trained using *AdaBoost* learning algorithm Matas & Sochman (Oct. 2008) and can be calculated very fast using *integral image*. In Choi (2006) real time on-road vehicle detection with optical flow and Haar-like feature is presented. The algorithm detects vehicles moving in the same direction up to 88% of accuracy. Fast vision-based pedestrian detection using Haar-like features is presented in Monteiro et al. (2006). Result shows that system can detect pedestrian at 17 frames per second on 1.7 GHz processor with pedestrian detection rate up to 90% of accuracy. Thorough review of image processing methods in visual navigation for mobile robots is given in Bonin-Font et al. (2008).

Additional estimators are used in order to increase robustness and reliability of a system based on vision system in the loop. A vision-based range estimator for leader-follower based on Immersion and Invariance is presented in Morbidi et al. (2010). The proposed reduced-order nonlinear observer is simple to implement, easy to tune and achieves global asymptotical convergence of the observation error to zero. The extended and unscented Kalman filters have been used in Mariottini et al. (2007) and Mariottini et al. (2005) to estimate the robots' relative distance.

In this paper we present an implementation of an algorithm for fast real-time visual tracking and localization in multi-agent system comprised of Wifibot mobile platforms Lab (2005). Mobile platforms, equipped with IP cameras, are moving in formation that is established and maintained by using a visual feedback. Leader agent is followed by follower agent which uses only images captured by its camera as feedback information.

The paper is organized as follows. In section 2 algorithm for fast object detection, using Haar-like features and integral image has been introduced followed by the description of strong classifiers and cascades of classifiers. Section 3 introduces the method for the platform position and orientation detection, using five differently trained cascades of classifiers. In section 4, the laboratory setup for platform tracking, comprised of two Wifibot platforms is described Design of Extended Kalman Filter is presented in section 5, while experimental results on platform localization are given in section 6.

2. Fast visual detection algorithm

2.1 Features and integral image

The algorithm for visual detection and localization of mobile agents, described in this article, is based on the method called "rapid object detection using boosted cascade of simple features", introduced by Viola and Jones Viola & Jones (2001). This algorithm classifies images based on the value of simple upright features made up of white and gray rectangles. The value of proposed features is defined as the difference between the sum of the image pixels within white and gray regions. In order to improve the efficiency of the algorithm, Leinhart and Maydt Lienhart & Maydt (2002) have extended the basic set of haar-like features by the set of 45° rotated rectangles. The extended set of features that has been used in the proposed object detection system is shown in Fig 1.
The value of a feature can be calculated by

$$feature_I = \sum_{i \epsilon I = \{1,\ldots,N\}} \omega_i \cdot RecSum(r_i), \tag{1}$$

where $RecSum(r_i)$ denotes pixel sum within *i*-th feature rectangle, $\omega_i \epsilon \mathbf{R}$ represent the weights with opposite signs, used for compensation of the differences in area size between two rectangles, and N is the number of rectangles the feature is composed of. In Lienhart & Maydt (2002) a method for fast calculation value of upright and rotated features at all scales at constant time has been proposed. Authors proposed an intermediate representation for the image called the *integral image* Viola & Jones (2001). The method is based on two auxiliary images - *Summed Area Table* $SAT(x, y)$ for upright rectangles and *Rotated Summed Area Table* $RSAT(x, y)$ for rotated rectangles. $SAT(x, y)$, defined by

$$SAT(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y'), \tag{2}$$

can be calculated with one pass over all pixels in the image and it represents the sum of the pixels of the upright rectangle with top left corner at $(0, 0)$ and bottom right corner at (x, y).

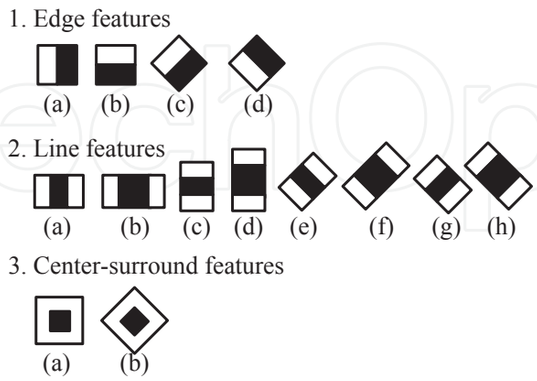


Fig. 1. Feature prototypes of simple Haar-like and center-surround features used for the platform detection

On the other hand, for the calculation of $RSAT(x, y)$, defined by

$$RSAT(x, y) = \sum_{x' \leq x, x' \leq x - |y - y'|} I(x', y'), \tag{3}$$

two passes over all pixels in the image are needed. $RSAT(x, y)$ gives the sum of pixels within rectangle rotated for 45° having the most right corner at (x, y). Once equations (2) and (3) have been calculated, only four lookup tables and three arithmetic operations are required to determine pixel sum of any upright or 45° rotated rectangle. Accordingly, the difference between two rectangle sums can be determined in maximally eight lookup tables and six arithmetic operations.

2.2 Simple and strong classifiers

Having defined the feature set, the training set of positive images (containing the object of interest) and the training set of negative images (containing no object), it is necessary to construct a classification function (classifier) that separates positive from negative images. A classifier can consist of one or more features. A classifier consisting of only one feature is also called a week classifier and it is defined by

$$h_j = \begin{cases} 1 & if \quad p_j f_j(x) < p_j \theta_j \\ 0 & otherwise \end{cases} \tag{4}$$

where f_j is a feature, θ_j a threshold, p_j parity indicating the direction of the inequality sign and x represents the image being classified. Because of their low accuracy, which is often within the range of 50-80%, single-featured classifiers are not very useful in practice. Much better detection performance provides a *strong* classifier that is based on a set of features selected in training procedure using *AdaBoost* algorithm described in Matas & Sochman (Oct. 2008) and Viola & Jones (2001). It trains a desired number of weak classifiers for the given image and forms a more complex classifier made as a linear combination of them, namely:

$$h(x) = \begin{cases} 1 \ \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 \qquad\qquad otherwise \end{cases} \tag{5}$$

Since the detection performance is often strict, it results in a strong classifier, which in order to be executed, requires computing a large number of features. This significantly increases the computational time and makes it inapplicable in real time problems.

2.3 Cascade of classifiers

This problem can be avoided, by construction of a *cascade* of classifiers providing an increased detection performance Viola & Jones (2001). Block diagram of the detection cascade is shown in Fig. 2. The cascade comprises several interconnected stages of classifiers. The detection process starts at first stage that is trained to detect almost all objects of interest and to eliminate as much as possible non-object patterns, i.e. classifiers that detect distinctive features of the object that are often describable using only few of the features from the used feature set (Fig. 1). The role of this classifier is to reduce the number of locations where the further evaluation must be performed. Every next stage consists of more complex classifier, trained to eliminate negative patterns being admitted through previous stage. If the object of interest has been detected at one stage, the detection process continues at the next stage. Otherwise the sub-window being checked is classified as a non-object and discarded immediately. The overall outcome of cascade is positive only in case the object has been detected by all stages.

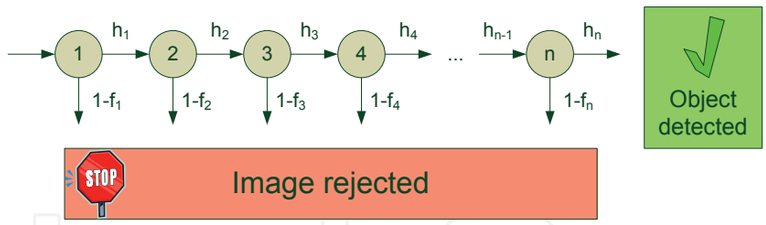


Fig. 2. Block diagram of the detection cascade with n stages

The use of a cascade significantly reduces computation time since the most patterns being tested are negative and thus discarded at lower stages composed of simpler classifiers, thus providing that the evaluation of higher stages occurs only on object-like regions. Every stage of cascade is characterized by hit rate h and false alarm rate f. Accordingly, the overall hit rate of cascade is h^n, and false alarm rate f^n, where n represents the number of stages. Overall, using several stages that have very high hit rate and relatively low false alarm rate results in a cascade with a high overall hit rate and a very low overall false alarm rate, since both overall values are obtained by multiplying individual rates.

3. Platform localization

3.1 Platform detection

Robot localization method presented in this paper identifies the platform location and orientation with respect to the camera, by image analysis using different cascades of classifiers. The main idea is to train several cascades in a way that each cascade is trained on a set of images taken from a specific angle of view. Thereafter, the platform detection is performed by applying all cascades on each frame captured from a Wifibot camera installed on a platform. Information about platform position and orientation can be then obtained by analyzing outputs of all cascades, as they depend on platform orientation. For example, in case detection is performed on a frame containing Wifibot platform with sideways view, it is expected that the best detection results would be obtained by cascade trained to detect platform from this angle of view. It should be noted that the resolution of orientation detection depends on the number of cascades being applied - higher resolution requires more cascades. However, the final number of cascades has to be selected as a trade off between the desired resolution and available computation time.

Since there are no significant differences in the front and the back view of the platform, as well as in the left and the right view, the algorithm for calculation of the angle of view has two solutions, which makes the orientation detection procedure more complicated. This problem has been solved through coloring Wifibot batteries in different colors, taking into account that the grayscale representations of both colors remain unchanged, so these colors do not influence grayscale images used in training process. Additional orientation information in form of different battery colors is obtained in the following way. The method utilizes the fact that an opposite platform view results in an opposite battery order with respect to the left image side. For example, considering that the left battery is colored red and the right battery green, their order in front platform view will be red - green, while in back view it will be green - red. Therefore, locations of batteries in an image has to be detected first, which is accomplished by detecting their colors. Having known the battery locations, their order with respect to the left image side becomes also known. In case of sideways platform view,

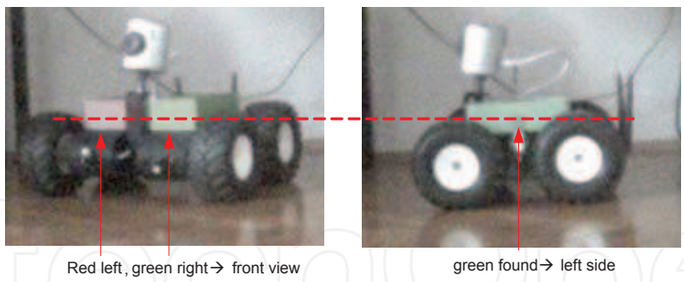


Fig. 3. Solving of the platform symmetry problem

only one battery color is detected depending on which side the platform is seen from. Need for battery color detection makes no significant influence on the overall algorithm execution time. Described strategy for solving the problem of angle of view ambiguity is demonstrated in Fig. 3.

3.2 Cascades training

Careful selection of images needed for training cascades is of special importance for accurate detection of the predefined object. In an effort to obtain a desired system robustness, 110 images containing Wifibot platform under different lighting conditions have been used (Fig. 4(a)). All images were taken by a camera mounted on the platform. During image collection process, the platform was located at different distances and orientations with respect to the platform being photographed. Once the set of positives has been created, only those image regions containing the Wifibot platform have been extracted (Fig. 4(b)). Dimension of extracted region was set to 50x22 pixels. The number of positives has been increased to 4000 by deriving new images characterized by different contrast and rotation angle as shown in Fig. 5. The set of negative images comprised of 5000 samples without Wifibot platform.
The number of cascades used in the system has been set to five. For that reason, the base of positive images has been eye-checked and sorted to five new subsets, each of them containing images of the platform taken from the specific angle of view. Due to the platform symmetry, the process of positive image sorting has been done according to two main angles of view - front-back and sideways. Accordingly, two cascades of classifiers were trained for detection of the platform from this two directions - *Wifi_Front_Back* and *Wifi_Side*. In order to enhance the quality of the orientation detection, two additional cascades of classifiers that provide

(a)

(b)

Fig. 4. Examples of positives, a) images taken by Wifibot camera, b) positive samples created from images

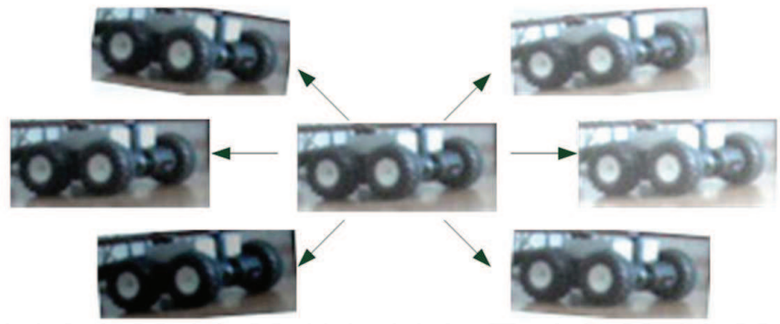


Fig. 5. Deriving of additional positives

detection of the platform from more specified angles were used. One of them, named *Wifi_FB_Extreme* is intended for detection of the platform from strictly front-back view, while *Wifi_Side_Extreme* detects the platform from strictly side view. The last cascade - *Wifi_All* has been trained for the platform detection from any angle of view, thus providing more robust detection of the platform location in an image. Angles of view covered by described cascades are shown in Fig. 6. Training of all cascades has been performed using *HaarTraining* application from *OpenCV* library *OpenCV On-line documentation* (Oct. 2008).

3.3 Searching method

Searching for the platform in current frame is carried out using resizable *searching window* that defines the image area being checked. Three different searching windows can be seen in Fig. 7. When platform detection is executed within these three windows, the platform will be detected only within red-colored searching window. The sides ratio of a searching window is always kept at constant value defined by sides ratio of classifiers used in cascades. Process of searching for the platform starts within the largest searching window that satisfies defined sides ratio and is initially located at the top left image corner. After all five cascades of classifiers have been applied to this image area, the detection continues within other image areas as the searching window shifts through the image until it reaches the right bottom image

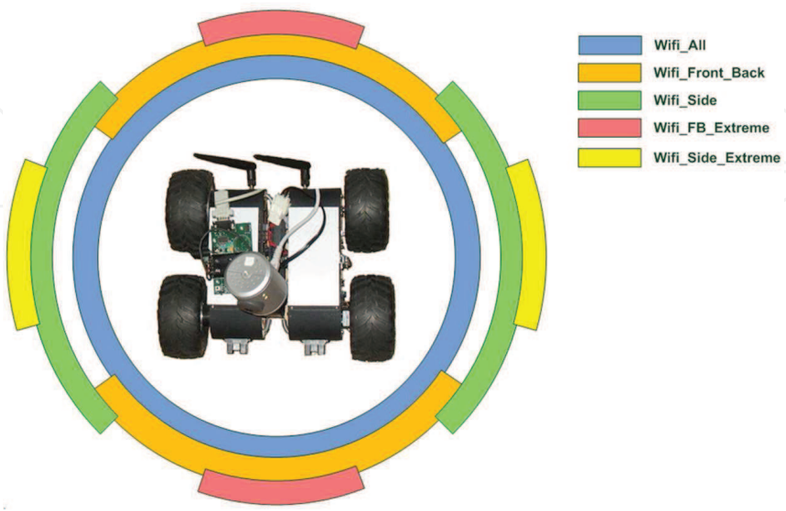


Fig. 6. Angles of view covered by five differently trained cascades

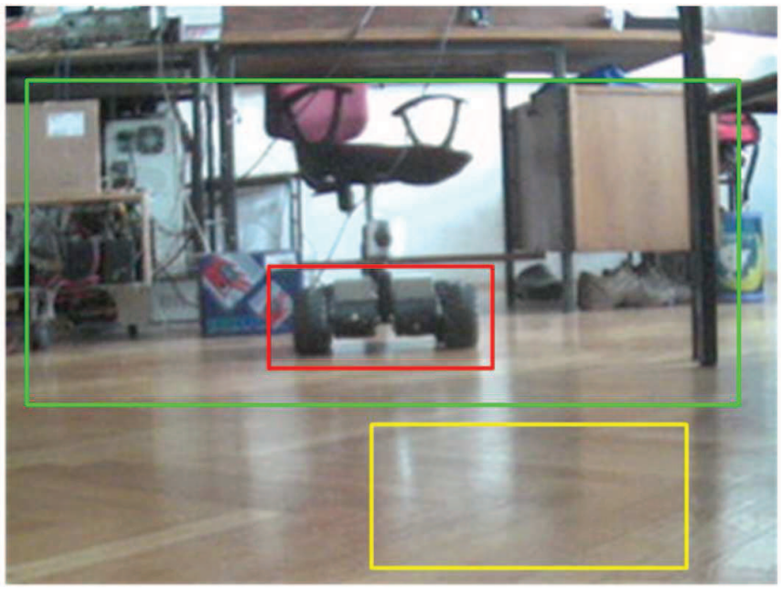


Fig. 7. Three different searching windows

corner. When the whole image has been checked, searching window is resized by predefined *scale factor* and the described procedure is repeated until the searching window reaches predefined *minimum size*. The searching algorithm is very fast, since a feature calculation based on integral image has been used. Described searching method, involving the whole image checking, is used only at the beginning of detection process in order to detect initial platform position. Once detected, the platform is being tracked by searching window which size can vary within 20% of the size of the window that detected the platform in previous frame. A scale factor, used for window resize, has been set to 0.95. Furthermore, this searching window is applied only to a narrow image area close to the last detected platform location. In this way, the speed of searching algorithm has been additionally increased. The algorithm was able to process 15 frames per second on 3.0 GHz Pentium IV processor with 1GB RAM.
Fig. 8 shows the screen of an application developed for validation of a single cascade using static images. After the platform detection has finished, this application highlights all searching windows the platform was detected within. As can be seen, usually there are more than one hits, closely located one to each other, thus called *neighbors*. When the number of neighbors is greater than the predefined *min_neighbours* parameter, the overall detection result is positive, and the platform location in image is highlighted by the representative searching window (blue rectangle in Fig. 8). Generally, all five cascades of classifiers are used in searching procedure which results in five different numbers of neighbors, one for each cascade. Fig. 9 shows the number of neighbors obtained by applying all five cascades to live video stream containing a static sideways view of a Wifibot platform. As can be noticed, each cascade has detected a certain number of neighbors. Moreover those numbers vary as the time is running, due to variations in lightning conditions and camera noise. As expected, the most neighbors have been detected by *Wifi_Side* and *Wifi_Side_Extreme* cascades. Decision making part of described orientation detection system selects a cascade that has detected the most neighbors in current frame. In case the number of neighbors is grater then the predefined *min_neighbours* parameter, selected cascade indicates the angle of view, i.e. current platform orientation with respect to the camera, according to the map shown in Fig. 6. There is one

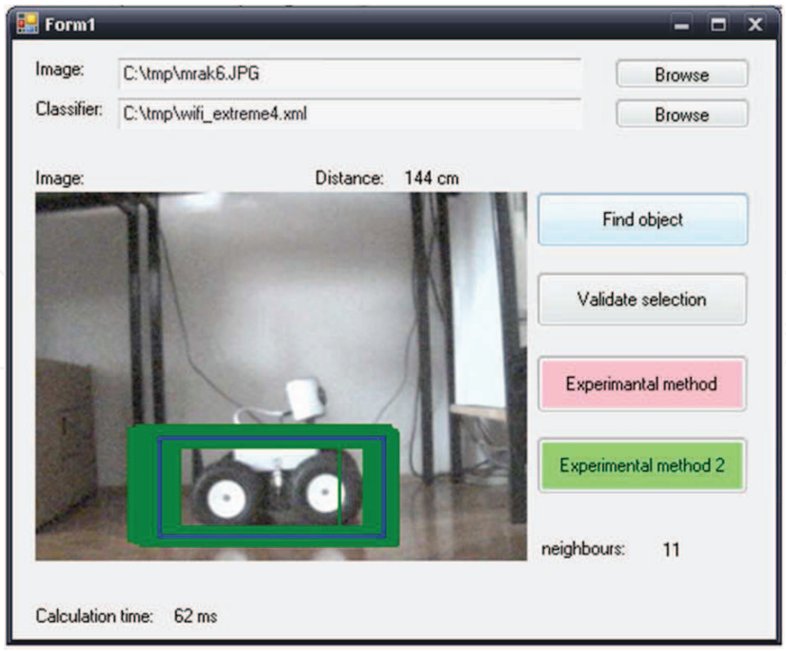


Fig. 8. Windows application for cascade validation on static images

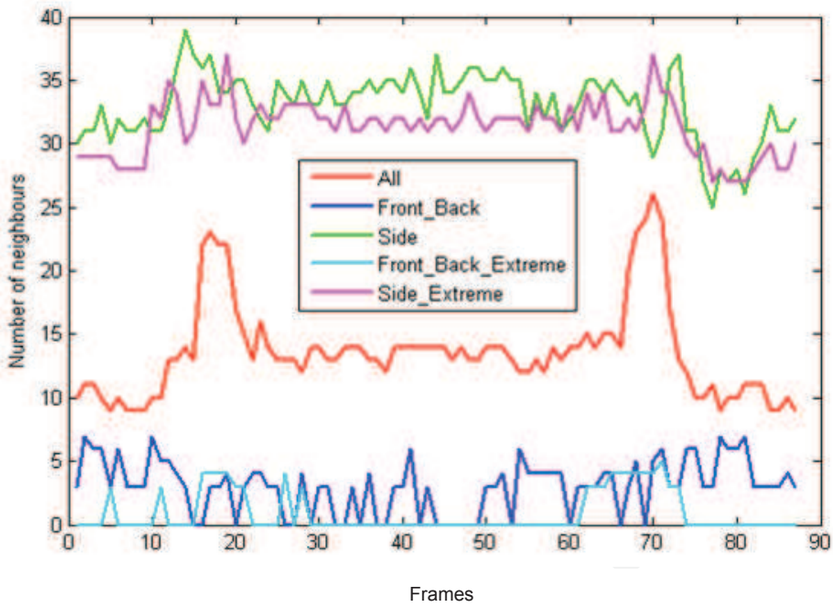
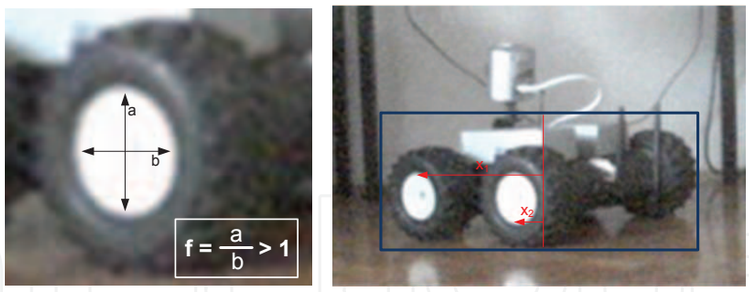


Fig. 9. Number of neighbors obtained by each cascade on a live video stream containing a sideways view of a Wifibot platform

supplementary condition selected cascade must satisfy in order to be declared as a hit - the percentage of its hits in the last ten frames must be greater than the experimentally adjusted parameter *min_percentage*. This condition provides a smooth transition in cascades outputs as the platform rotates.



(a) (b)

Fig. 10. a) axis length ratio, b) wheel distances from the platform center

In order to enhance the quality of orientation detection performed by cascades, an additional image analysis based on the calculation of wheel parameters has been applied. These parameters describe two white elliptic regions within platform wheels, which can be easily detected within positive searching window. One parameter is axis length ratio f (Fig. 10(a)), and the other two parameters $x1$ and $x2$ represent the wheel distances from the platform center (Fig. 10(b)). The angle of view is unambiguously defined by these parameters, and thus can be precisely reconstructed.

4. Platform tracking setup

Described visual-based system for platform localization has been tested on laboratory setup comprised of two Wifibot platforms - leader and follower. The follower's task is to track the remotely controllable leader that performs an arbitrary trajectory, while keeping the predefined inter-distance. The designed tracking control system has been divided into two parts - platform inter-distance control subsystem and subsystem for control of follower orientation with respect to the leader. The subsystem for follower orientation control provides actions needed for follower to face the leader, which is accomplished when the leader is located in the center of image captured by the follower. The distance between the leader and the follower is obtained from the leader size in current frame. An increase in the platform inter-distance causes the decrease in the platform size in an image, and vice versa. This dependency has been experimentally obtained. Once cascades detect position of the platform a distance from camera is calculated based on the width of the frame determined from neighbors (blue rectangle in Figure 8). A function of distance with respect to width is depicted in Fig. 11. Blue dots represent measurements while red line is interpolation of the following form:

$$d(w) = 50850 \cdot e^{-0.07287 \cdot w} + 311.2 \cdot e^{-0.005298 \cdot w} \tag{6}$$

It is evident that relation between width of an object and its distance from a camera is not linear which is characteristic of human eye as well.

The difference between current and desired platform inter-distance causes the follower forward/backward motion, while the leader's displacement from the image center causes the follower rotation until the leader appears in the image center. Block diagram of the designed control system, containing two described subsystems, is shown in Fig. 12. The control

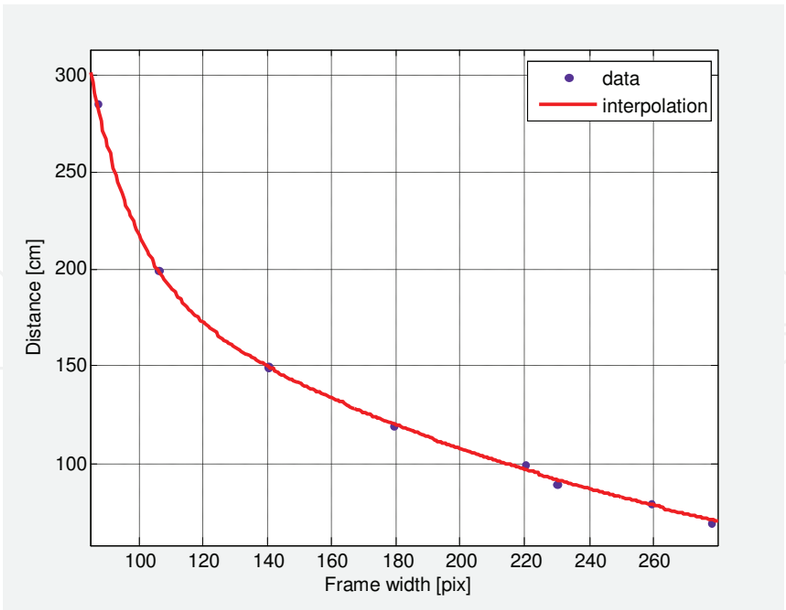


Fig. 11. Distance from camera with respect to the frame width

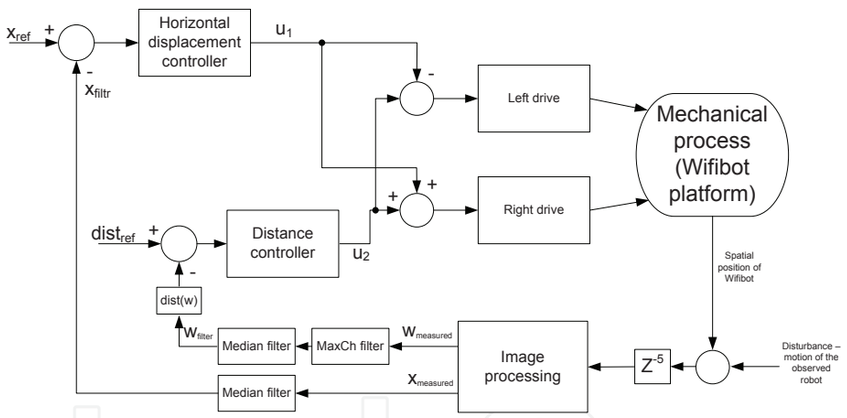


Fig. 12. Block diagram of the system for platform following

algorithm with sample time of Td=50 ms, has been executed on 3.0 GHz Pentium IV processor, while the camera images and commands for platform motion have been transmitted trough the wireless network established between PC and access point embedded in the platform. Due to the noise in angle and distance measurement (Fig. 13) Median and MaxChange filters have been added in the feedback loop. For correct design of tracking controllers an additional delay of 250 ms (5xTd), caused by image capture and processing by the camera on board of the platform, is introduced in the feedback loop. Proportional controllers have been used for both - distance and orientation control. It is important to note that the horizontal displacement estimation of a dynamical object using low angle of view and low resolution camera is difficult due to high noise amplification, specially for distant objects that appear small compared to the

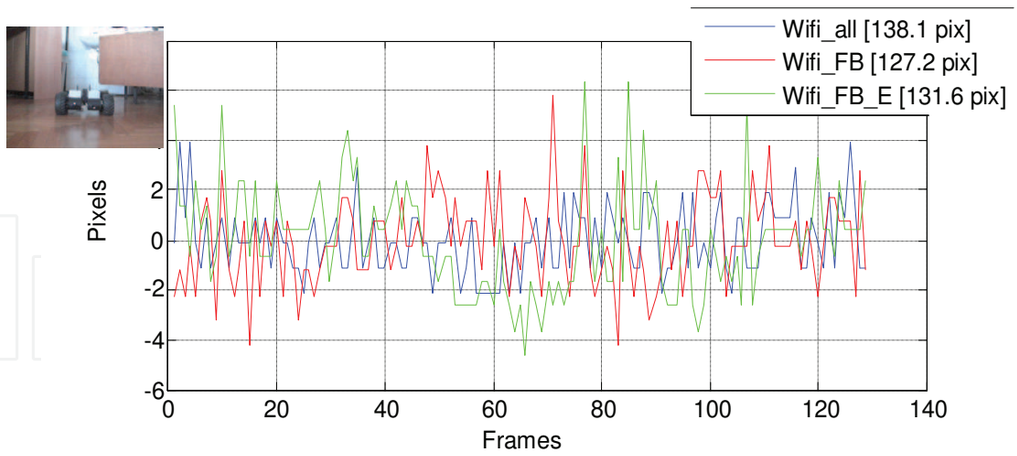


Fig. 13. Noise in calculated distance ($wavg$=133 pix, d=160 cm)

picture frame and the noise amplitude of only one pixel represents a very high deviation in the physical displacement of the object.
The estimate of this relation is written in (7).

$$\frac{\Delta x}{\Delta pix} \approx \frac{37}{w}\text{cm} \tag{7}$$

This phenomena can also be observed in the Fig. 11 where the curve tangent becomes very steep for large distances. Also, detecting horizontally moving objects, which are close to the camera, tends to be difficult since they fully appear on the picture frame just for a fraction of time which is comparable to the camera delay. Because of that the usage of unfiltered signals from the vision system in the conventional control schemes generates big overshoots and chattering in controlled signals when tracking very distant or very close objects. Model based filtering is a solution to this problem.

5. Kalman filter design

5.1 Tracked agent dynamics

Synthesis of a control scheme based on the measurements from a visual tracking system used in an environment with changing lighting conditions and obstacles, which very often cover parts of the tracked object, becomes very difficult due to often unavailability of sensor data. Furthermore, the available data is often noisy due to low camera resolution and stochastic nature of the used detection scheme. For this reason, a good estimator and filter is needed in the control loop. In this application, we have used a common technique, namely an *extended Kalman filter* Mohinder S. Grewal (2008). Since no information other than vision is available for the tracked agent, it's distance and horizontal displacement has been modeled by an integrator with an unknown gain, i.e.:

$$\begin{bmatrix} \Theta(k) \\ v_D(k) \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Theta(k-1) \\ v_D(k-1) \end{bmatrix} \tag{8}$$

where Θ represents the object distance D or horizontal displacement x, v_D the object speed in either horizontal or vertical direction and T the discretization period of the control scheme. Also, inverses of (6) and (7) have been used as state measurement functions.

5.2 Host agent dynamics

In the multi-agent systems, both the tracked agent and the host agent are in motion, meaning the host agent motion also influences the tracked agents frame position making the model (8) imprecise. That is why model (8) has to be updated with the host motion profile which in the case of the Wifibot is directly available from the incremental encoders mounted in the wheels. Availability of the direct wheel speed measurements eliminates the need for accurate host robot dynamical model. With host object motion taken into account, the agent distance is modeled with:

$$\begin{bmatrix} D(k) \\ v_D(k) \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} D(k-1) \\ v_D(k-1) \end{bmatrix} + \begin{bmatrix} -2 \cdot T \\ 0 \end{bmatrix} \cdot \bar{N}_{enc}(k), \tag{9}$$

where $\bar{N}_{enc}(k)$ represents a mean value of left and right incremental encoder outputs, namely: $\bar{N}_{enc} = 0.5 \cdot (N_{encL} + N_{encR})$. Furthermore, making a complete dynamical model of the tracked object's horizontal displacement requires knowledge of the host robot dimensions together with left and right incremental encoder outputs.

From Fig. 14 we have: $R = \sqrt{\left(\frac{d}{2}\right)^2 + a^2}$, which gives a relation for the angular velocity of the host agent:

$$\omega = K \cdot \frac{N_{encR} - N_{encL}}{R} \cdot \frac{d}{2R} = \frac{d}{R^2} \cdot \Delta N_{enc}, \tag{10}$$

where ΔN_{enc} represents the difference between the encoder outputs and K is the encoder gain. The small angle approximation $\dot{x} \approx D \cdot \omega \approx \frac{D \cdot d}{R^2} \cdot \Delta N_{enc}$ is then used to obtain the final tracked agent horizontal displacement model:

$$\begin{bmatrix} x(k) \\ v_x(k) \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x(k-1) \\ v_x(k-1) \end{bmatrix} + \begin{bmatrix} \frac{D \cdot d}{R^2} \cdot T \\ 0 \end{bmatrix} \cdot \Delta N_{enc}(k) \tag{11}$$

Extended Kalman filters based on dynamical models (9) and (11) with inverses of (6) and (7) as state measurement functions are then used as filters to improve signal to noise ratio of the vision system and as the estimators of the tracked object position when there is no data available.The experimental data follows in the next section.

6. Experimental results

Performance of the proposed localization and tracking method has been tested on the experimental setup comprised of two Wifibot platforms in leader-follower formation. First experiment has been conducted on static platform in order to investigate quality of all five cascades.Results obtained by the experiment are presented in Fig. 15(a). Four different postures (angles of view) of Wifibot platform on three different distances have been examined. For each posture and distance (12 cases) 90 frames have been processed (15 fps). In case of front-back angle of view (images 1, 2 and 3) results are very good. Two cascades, *Wifi_All* and *Wifi_Front_Back*, have 100% accuracy. Slight deviation can be noticed with cascade *Wifi_FB_Extreme* in case platform is far away from the camera; only 5.6% accuracy is achieved. However, this result is also useful since *Wifi_Side_Extreme* for the same case is 0%. If one compares *Wifi_Front_Back* with *Wifi_Side* and *Wifi_FB_Extreme* with *Wifi_Side_Extreme*, then for all three images *Wifi_Front_Back* and *Wifi_FB_Extreme* cascades are dominant (presented gray in Fig. 15(a)). For side view (images 4, 5 and 6) cascades *Wifi_Side* and *Wifi_Side_Extreme* are dominant with 100% accuracy which is exceptionally good result. Results obtained

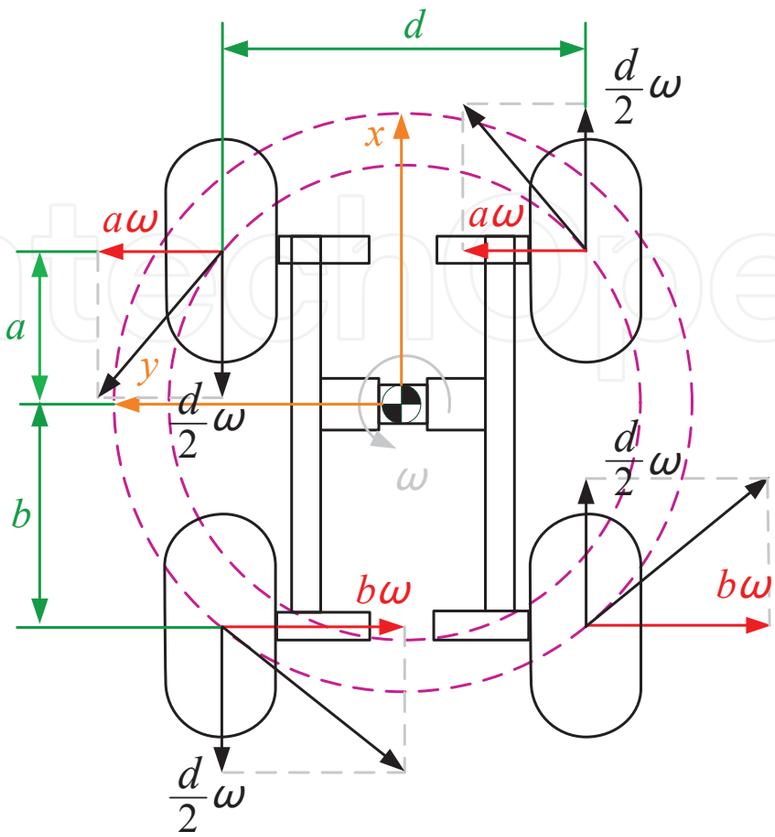
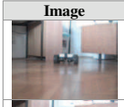


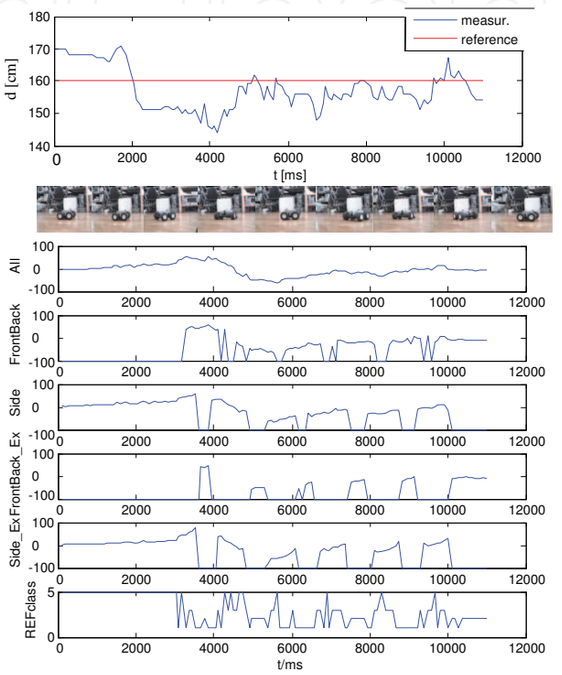
Fig. 14. Wheel positions and dimension of the Wifibot robot ($a = b$)

for conditions when angle of view was somewhere between front-back and side view are demonstrated on images 7 to 12. Although situation is not as clear as for previous examples, in most cases *Wifi_Front_Back* and *Wifi_Side_Extreme* cascades are dominant, which demonstrates that platform is inclined.

Second set of experiments have been performed on moving platform. The leader executes a circular movement (small radius) while follower has been commanded to remain at distance of 160 cm. Results, depicted in Fig. 15(b), are comprised of measured distance (top of the figure), displacement of the leader from the center of image (calculated by cascades) and reference cascade (bottom of the figure). As may be seen the follower keeps the distance error within ś10 cm which is less than 10% of the set point value. In the same time horizontal displacement control algorithm rotates the follower so that leader remains within limits of ś50 pixels from the center of image. Depending on the angle of view, particular cascade detects the leader and calculates its displacement: cascade *Wifi_All* is active all the time during experiment, *Wifi_Side* and *Wifi_Side_Extreme* detect the leader at the beginning of experiment as it moves sideways, while at the same time *Wifi_Front_Back* and *Wifi_FB_Extreme* remain inactive. In 3^{th} second of experiment the leader starts to turn which is immediately recognized by *Wifi_Front_Back* cascade that becomes active together with *Wifi_Side* and *Wifi_Side_Extreme*. Activation and deactivation of other cascades can be easily tracked throughout experiment.

Image	*ALL*	*FB*	*SIDE*	*FB_E*	*SD_E*	*No.*
	1.0	1.0	0.0	0.056	0.0	1
	1.0	1.0	0.0	1.0	0.0	2
	1.0	1.0	0.056	1.0	0.0	3
	1.0	0.26	1.0	0.0	1.0	4
	0.68	0	1.0	0.0	1.0	5
	1.0	0.39	1.0	0.011	1.0	6
	1.0	0.95	0.0	0.0	0.0	7
	1.0	1.0	0.32	0.0	0.033	8
	1.0	1.0	0.0	0.0	0.38	9
	0.95	0.31	0.088	0.0	0.011	10
	0.99	0.32	0.24	0.0	0.39	11
	0.93	0.23	1.0	0.0	1.0	12

(a) Classification results (static images)



(b) Tracking of a circular movement of the leader

Fig. 15. Experimental results

Diagram *REFclass* shows which cascade is dominant at particular moment: 1 = *Wifi_All*, 2 = *Wifi_Front_Back*, 3 = *Wifi_Side*, 4 = *Wifi_Front_Back*, and 5 = *Wifi_Side_Extreme*. The video clips displaying described experiments can be downloaded from http://flrcg.rasip.fer.hr/movies/. Third set of experiments have been performed to test the proposed extended Kalman filter. Figures 16 and 17 show tracked agents horizontal displacement and distance signals with and without using a Kalman filter.

Covariance matrices are set to $P_0 = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$, $Q = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$ and $R = 10$. An improvement of tracking performance can be seen, specially for agents which are distant from the camera and which appear small in the detected picture. In the final setup, maximal change and median filters from the control loop Fig. 12 have been replaced with the proposed Kalman filters.

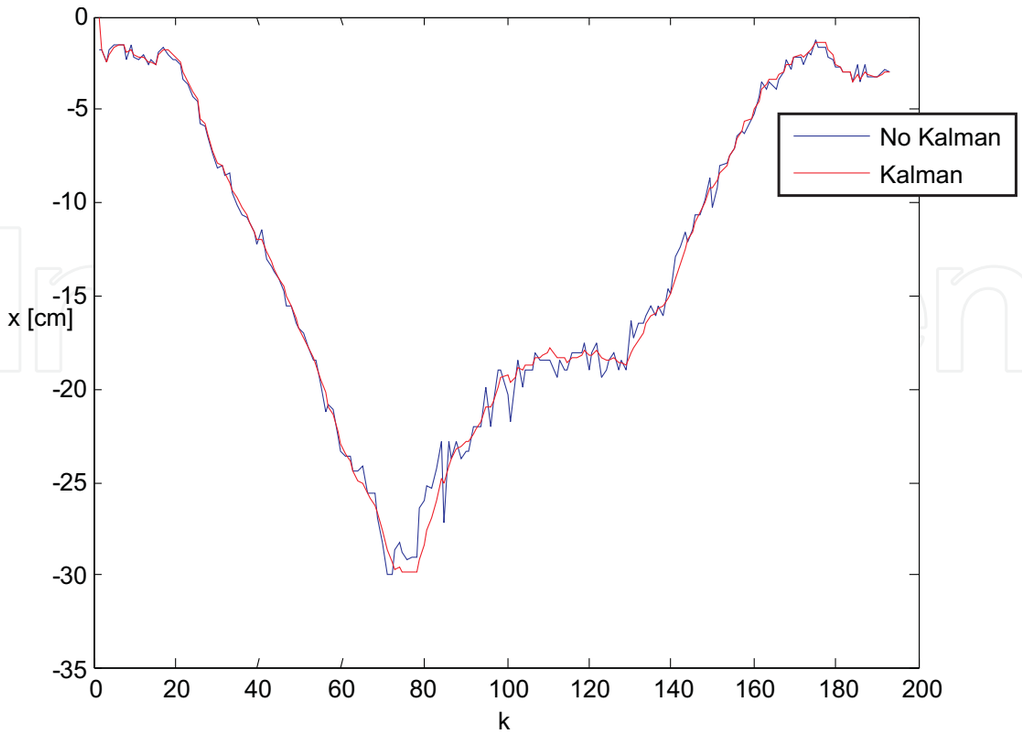


Fig. 16. Horizontal displacement of the tracked agent at distances shown in Fig. 17

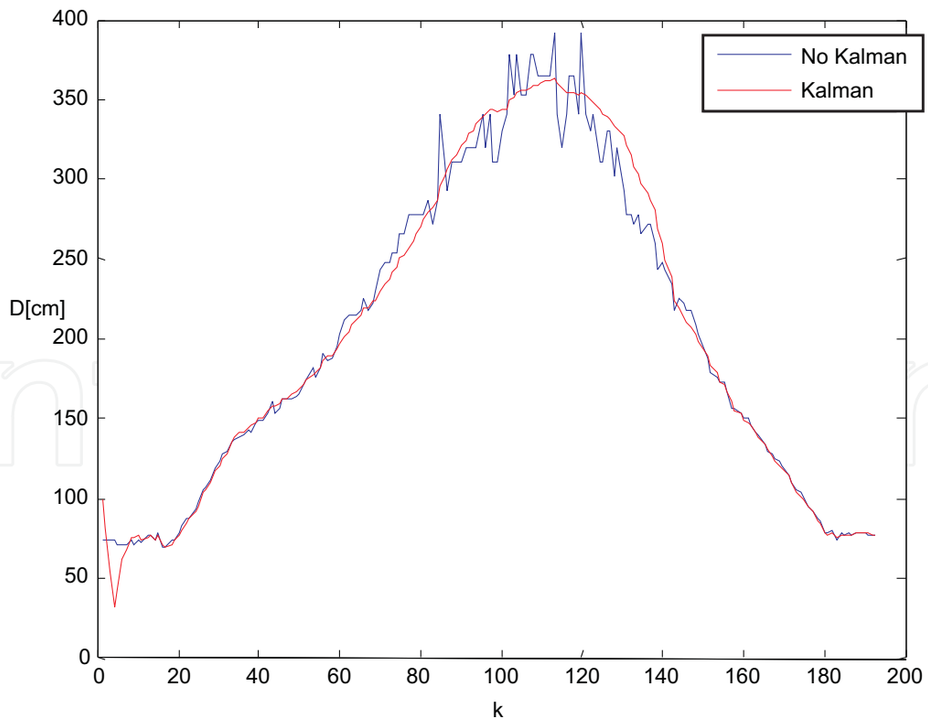


Fig. 17. Distance of the tracked agent

7. Conclusion

This paper presents localization and tracking strategy that is based on an extremely rapid method for visual detection of an object, thus providing a real time solution suitable for the design of multi-agent control schemes. The agents tracking and localization is carried out through five differently trained cascades of classifiers that process images captured by cameras mounted on agents. A training of 5 cascades, described herein, has employed ŞpositiveŤ and ŞnegativeŤ images, i.e. images with and without an object that should be recognized. Once trained, the cascade have been used for classification process (based on Haar-like features), that enables each agent to determine its relative position and orientation with respect to all other agents performing tasks in its field of view. Performance of the proposed method has been demonstrated on a laboratory setup composed of two mobile robot platforms. Experimental results demonstrate that overall accuracy of recognition is very good (in some case even 100%), thus providing accurate calculation of distance between the leader and the follower. Although acceptable, results of calculation of displacement from the center of image are reticent, and can be improved by using model based estimation, i.e. proposed extended Kalman filters. Further improvements can be made with the usage of a higher image resolution camera. Future work will be toward improvement of robustness of the proposed method with respect to various environment conditions (light intensity, shadows, etc).

8. References

Bonin-Font, F., Ortiz, A. & Oliver, G. (2008). Visual naviation for mobile robots: A survey, *Journal of Intellient & Robotic Systems* 53.

Chiem, S. Y. & Cervera, E. (2004). Vision-based robot formations with bezier trajectories, *Proceedings of Intelligent Autonomous Systems* pp. 191Ű–198.

Choi, J. (2006). Real time on-road vehicle detection with optical flows and haar-like feature detector, *a final report of a course CS543 Computer Vision* .

Cruz, D., Mcclintock, J., Perteet, B., Orqueda, O. A. A., Cao, Y. & Fierro, R. (2007). Decentralized cooperative control - a multivehicle platform for research in networked embedded systems, *Control Systems Magazine, IEEE* 27(3): 58–78.
URL: *http://dx.doi.org/10.1109/MCS.2007.365004*

Dani, A. P., Gans, N. & Dixon, W. E. (2009). Position-based visual servo control of leader-follower formation using image-based relative pose and relative velocity estimation, *ACC'09: Proceedings of the 2009 conference on American Control Conference*, IEEE Press, Piscataway, NJ, USA, pp. 5271–5276.

Gans, N.R.; Dani, A. D. W. (2008). Visual servoing to an arbitrary pose with respect to an object given a single known length, *ACC'08: Proceedings of the 2008 conference on American Control Conference*, pp. 1261 – 1267.

Jennings, J., Whelan, G. & Evans, W. (1997). Cooperative search and rescue with a team of mobile robots, *Advanced Robotics, 1997. ICAR '97. Proceedings., 8th International Conference on* pp. 193–200.

Lab, W. (2005). Wifibotsc datasheet, *Hardware Manual* .

Lienhart, R. & Maydt, J. (2002). An extended set of haar-like features for rapid object detection, Vol. 1, pp. I–900–I–903.
URL: *http://dx.doi.org/10.1109/ICIP.2002.1038171*

Mariottini, G. L., Prattichizzo, D. & Daniilidis, K. (2005). Vision-based localization of leader-follower formations, *IEEE Conference on Decision and Control and European Control Conference*, pp. 635–640.

Mariottini, G., Morbidi, F., Prattichizzo, D., Pappas, G. & Daniilidis, K. (2007). Leader-follower formations: Uncalibrated vision-based localization and control.

Matas, J. & Sochman, J. (Oct. 2008). Adaboost.
URL: *http://www.robots.ox.ac.uk/ az/lectures/cv/adaboost_matas*

Mohinder S. Grewal, A. P. A. (2008). Theory and practice using matlab 3rd edition, *A John Wiley & sons, Inc* .

Monteiro, G., Peixoto, P. & Nunes, U. (2006). Vision-based pedestrian detecion using haar-like features, *Robotica* .

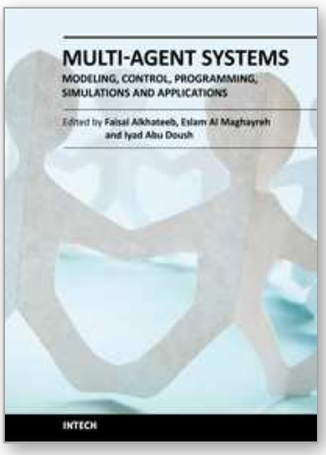
Morbidi, F., Mariottini, G. L. & Prattichizzo, D. (2010). Brief paper: Observer design via immersion and invariance for vision-based leader-follower formation control, *Automatica* 46(1): 148–154.

OpenCV On-line documentation (Oct. 2008).
URL: *http://opencvlibrary.sourceforge.net/*

Renaud, P., Cervera, E. & Martiner, P. (2004). Towards a reliable vision-based mobile robot formation control, *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on* 4: 3176–3181.

Tomono, M. (2006). 3-d object map building using dense object models with sift-based recognition features, *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on* pp. 1885–1890.

Viola, P. & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features, Vol. 1, pp. I–511–I–518.
URL: *http://dx.doi.org/10.1109/CVPR.2001.990517*



Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications

Edited by Dr. Faisal Alkhateeb

ISBN 978-953-307-174-9
Hard cover, 522 pages
Publisher InTech
Published online 01, April, 2011
Published in print edition April, 2011

A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents. Multi-agent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve. Agent systems are open and extensible systems that allow for the deployment of autonomous and proactive software components. Multi-agent systems have been brought up and used in several application domains.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Andelko Katalenic, Ivica Draganjac, Alan Mutka and Stjepan Bogdan (2011). Fast Visual Tracking of Mobile Agents, Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications, Dr. Faisal Alkhateeb (Ed.), ISBN: 978-953-307-174-9, InTech, Available from: http://www.intechopen.com/books/multi-agent-systems-modeling-control-programming-simulations-and-applications/fast-visual-tracking-of-mobile-agents

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen